# ภาษาเครื่องสำหรับภาษาสูง ตอน 1

บุญเลิศ เสียมทัศนา

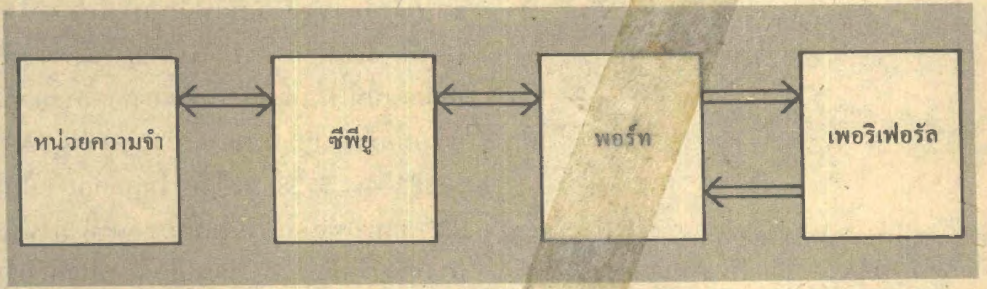
การเขียนโปรแกรมในภาษาสูง บางครั้งจำเป็นต้องสร้างโปรแกรมย่อยในภาษาเครื่องขึ้นมาใช้งาน อาจเป็นเพราะไม่สามารถเขียนโปรแกรมย่อยที่ต้องการในภาษาสูงนั้นได้ หรือเขียนได้แต่ต้องใช้เวลาในการปฏิบัติ (รัน) นานมากจนยอมรับไม่ได้ ความจำเป็นที่ต้องรู้ภาษาเครื่องจึงมีอยู่ โดยเฉพาะบนเครื่องไมโครคอมพิวเตอร์

อีกประเด็นหนึ่ง การศึกษาเพื่อเขียนโปรแกรมไม่ว่าในระดับใด ในปัจจุบันนิยมและผมก็เห็นด้วยที่ให้ศึกษาภาษาสูงที่เป็นโครงสร้างเช่นปาสคาลเสียก่อน แต่ก็เป็นปัญหาวัวพันหลักคือ ผู้ที่รู้ภาษาเครื่องมาก่อนมักจะทำความเข้าใจกับภาษาปาสคาลหรือซีได้เร็วกว่า ผมจึงเห็นว่าน่าจะหยิบเรื่องหลัก ๆ ของการเขียนโปรแกรมภาษาเครื่องมากล่าวถึง เพียงเพื่อให้สามารถเขียนโปรแกรมภาษาเครื่องง่าย ๆ ได้ และเพื่อเสริมสร้างความเข้าใจในภาษาสูง เช่น ในเรื่องค่าคงตัว ตัวแปร การผ่านค่า พอยน์เตอร์ในภาษาซี การจัดหน่วยความจำ และอื่น ๆ

ในการเขียนบทความชุดนี้ ผมตั้งใจจะให้ใช้ได้กับเครื่องทุกเครื่อง แต่เป็นไปไม่ได้ที่ผมจะยกตัวอย่างให้ทดลองโดยไม่อิงถึงซีพียูเบอร์หนึ่งเบอร์ใดโดยตลอด ในตอนต้นของบทความชุดนี้ ผมจึงจะกล่าวถึงซีพียูสมมติขนาด 8 บิทเพื่อให้เข้าใจง่าย ตอนต่อ ๆ ไปจะอ้างถึง 8088 และถ้าจะต้องไปเกี่ยวพันกับทางฮาร์ดแวร์ของเครื่อง จะหมายถึงไอบีเอ็ม พีซี ทั้งนี้เพราะผมกำลังจะนำเรื่อง "ลึกลงไปในไอบีเอ็ม พีซี" มากล่าวถึงเป็นอันดับต่อไป

## คอมพิวเตอร์

ผู้ที่เขียนโปรแกรมภาษาเครื่องส่วนมากมักจะมองคอมพิวเตอร์ในแง่ที่ประกอบด้วยส่วนสำคัญ 3 ส่วนคือ ซีพียู หน่วยความจำ และพอร์ท (port) ดังแสดงในรูปที่ 1 พอร์ททำหน้าที่เชื่อมต่อระหว่างซีพียูกับอุปกรณ์รอบข้าง เช่น คีย์บอร์ด จอภาพ เครื่องพิมพ์ โมเด็ม ฯลฯ จึงมีชื่อเรียกว่า Pheripheral Interface หรือ Pheripheral Adapter แต่มักจะนิยมเรียกกันสั้น ๆ ว่าพอร์ท ซึ่งสำหรับเรื่องพอร์ทจะเกี่ยวพันกับทางฮาร์ดแวร์ จึงขอพักไว้ก่อนและจะนำไปกล่าวในบทความชุดต่อไปดังกล่าวแล้ว




**รูปที่ 1 แสดงความสัมพันธ์ของอุปกรณ์ในระบบคอมพิวเตอร์**

สำหรับหน่วยความจำ มีหน้าที่เก็บคำสั่งและข้อมูล ขนาดของหน่วยความจำขึ้นอยู่กับความสามารถของซีพียูในการอ้างแอดเดรส เช่น ในซีพียูขนาด 8 บิท(คือมีขาดาต้า 8 ขา) หากมีขาแอดเดรส 1 ขา ก็จะอ้างแอดเดรสได้ 2 ตำบลที่ คือ 2 ไบท์ ได้แก่ ไบท์ที่ 0 กับไบท์ที่ 1 ถ้ามีขาแอดเดรส 2 ขา ก็จะอ้างแอดเดรสได้ 4 ไบท์ คือไบท์ที่ 00, 01, 10 และ 11 ในเลขฐานสอง หรือคือ 0, 1, 2 และ 3 ในเลขฐานสิบตามลำดับ

ถ้าซีพียูมีขาแอดเดรส 10 ขา ก็จะอ้างแอดเดรสได้ $2^{10}$ = 1024 ตำบลที่ แต่นิยมเรียกเป็น 1 KB มากกว่า (K หมายถึง กิโล) สำหรับซีพียูขนาด 8 บิทที่ผมจะยกมาเป็นตัวอย่างในตอนแรกนี้ ส่วนมากจะมีขาแอดเดรส 16 ขา จึงอ้างแอดเดรสได้ $10^{16}$ = 64 KB หมายถึงว่า เราจะจัดหน่วยความจำได้สูงสุดเพียงเท่านี้ แต่จะจัดเท่าไรขึ้นอยู่กับการออกแบบ ซึ่งเรื่องนี้เพียงแต่ให้รับทราบไม่จำเป็นต้องสนใจมากนัก เรื่องที่จำเป็นจะต้องสนใจคือ แบบของการอ้างแอดเดรส หมายถึงว่าเราจะอ้างแอดเดรสถึงหน่วยความจำแต่ละตำบลที่ในโปรแกรมได้อย่างไร ดังจะกล่าวต่อไป

ซีพียูในไมโครคอมพิวเตอร์มีชื่อเรียกเฉพาะว่าไมโครโปรเซสเซอร์ (Microprocesser) มองภายนอกจะเหมือนไอซีธรรมดาตัวหนึ่ง แต่ภายในจะประกอบด้วยอุปกรณ์ที่สลับซับซ้อนหลายอย่าง ซึ่งในขั้นต้นนี้ เรื่องที่เกี่ยวกับซีพียูขอให้สนใจในเรื่อง รีจิสเตอร์ ชุดคำสั่ง กับแบบของการอ้างแอดเดรส ก็เพียงพอ

## เลขฐานสิบหก

การเขียนโปรแกรมภาษาเครื่อง แต่ก่อนมีความจำเป็นจะต้องรู้เรื่องเลขฐานสองและฐานสิบหก ในปัจจุบันความจำเป็นนี้ดูจะลดลง แต่ถึงอย่างไรก็ควรจะทราบไว้บ้าง และในบทความนี้จะใช้เลขฐานสิบหกเป็นหลัก จึงขอนำมากล่าวเท่าที่จำเป็น

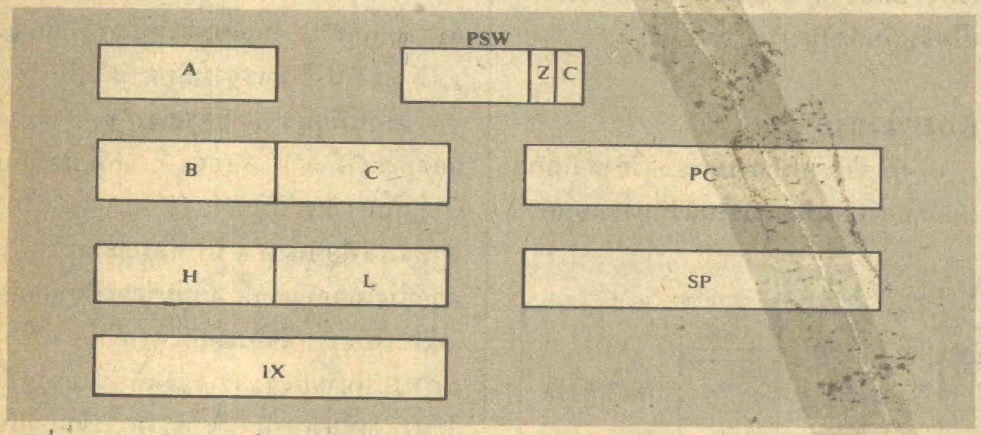
ในเลขฐานสิบเรามีตัวเลข 10 ตัว คือ 0 ถึง 9 เช่นกันในเลขฐานสิบหกก็จะมีตัวเลขอยู่ 16 ตัว คือ 0 ถึง 9 ต่อด้วย A ถึง F โดย A มีค่าเป็น 10 ในเลขฐานสิบ และ F มีค่าเป็น 15 ในเลขฐานสิบ การเขียนตัวเลขฐานสิบหกนิยมเขียนเป็น 2 แบบ คือขึ้นต้นด้วย $ หรือลงท้ายด้วย H เช่น $3E หรือ 3EH แต่ที่ลงท้ายด้วย H หากเลขตัวแรกเป็นเลขอักษรให้ขึ้นต้นด้วยศูนย์ เช่น $A6 เขียนเป็น 0A6H

ถึงแม้จะเป็นซีพียูจำลองก็ต้องกำหนดให้เหมือนของจริง ซึ่งความจริงผมเพียงแต่จับของเบอร์ต่าง ๆ มาผสมกัน คุณที่เคยผ่านภาษาเครื่องมาคงทราบดี แต่เรื่องทั้ง 3 เรื่องนี้ คือ รีจิสเตอร์ ชุดคำสั่ง และการอ้างแอดเดรส เป็นเรื่องที่พัวพันกันอยู่ คงจะต้องอ่านวนไปมา ผมจึงจะพยายามเขียนให้สั้น

### รีจิสเตอร์

รีจิสเตอร์คือหน่วยความจำซึ่งอยู่ในซีพียู ถ้าเป็นซีพียูขนาด 8 บิท รีจิสเตอร์จะมีขนาด 8 บิทด้วย แต่ก็มีซีพียูจำนวนหนึ่งที่จัดให้มีรีจิสเตอร์ขนาด 2 เท่า เช่นซีพียู 8 บิท มีรีจิสเตอร์ 16 บิท ซีพียู 16 บิทมีรีจิสเตอร์ 32 บิท เพื่อเพิ่มประสิทธิภาพให้แก่ซีพียูของตน ซีพียูจำลองก็เช่นกันดังแสดงในรูปที่ 2

รีจิสเตอร์ตัวสำคัญคือ แอคคิวมูเลเตอร์ (accumulater) ตามรูปที่ 2 คือรีจิสเตอร์ A




**รูปที่ 2 แสดงรีจิสเตอร์ในซีพียูจำลอง**

A, B, C, H และ L เป็นรีจิสเตอร์ขนาด 8 บิท ซึ่งสามารถนำมาต่อกันเป็น BC และ HL ทำให้เป็นรีจิสเตอร์ขนาด 16 บิท IX, PC และ SP เป็นรีจิสเตอร์ขนาด 16 บิท PSW เป็นรีจิสเตอร์ขนาด 8 บิท สำหรับเก็บแฟล็กแต่กำหนดให้ใช้เป็นแฟล็กเพียง 2 บิทเท่านั้นคือ แฟล็ก C และแฟล็ก Z ที่เหลืออีก 6 บิท ไม่ได้ใช้งาน

## ซีพียูจำลอง

ดังที่กล่าวแล้วว่าในตอนนี้ผมจะสมมติซีพียูขึ้นมาเบอร์หนึ่ง เพื่อใช้เป็นตุ๊กตาในการเขียนโปรแกรมภาษาเครื่อง การที่ผมไม่ใช้ซีพียูที่มีจริงก็เพื่อลดขั้นตอนในการเขียนโปรแกรม และเพื่อแสดงให้เห็นว่าในการเขียนโปรแกรมย่อยภาษาเครื่องเพื่อใช้ในภาษาสูงนั้น เรารู้เรื่องของซีพียูตามที่จะกล่าวต่อไปนี้ก็เพียงพอ

โดยทั่วไปแอคคิวฯจะทำหน้าที่เก็บข้อมูลเพื่อเป็นตัวตั้ง ในการดำเนินกรรมวิธีทางคณิตศาสตร์และคณิตศาสตร์ของบูลลีน เสร็จแล้วจะได้ผลในแอคคิวฯ เช่น จะบวกค่า 05H กับ 10H จะต้องให้ค่า 05H กับแอคคิวฯก่อน แล้วจึงบวกค่า 10H กับแอคคิวฯ ได้เป็น 15H ในแอคคิวฯ ซึ่งจะต้องนำไปดำเนินการต่อหรือเก็บในหน่วยความจำเพื่อรอดำเนินการต่อไป

ส่วนรีจิสเตอร์นอกนั้นแยกเป็น 2 ความคิด ของอินเทลจะใช้รีจิสเตอร์เป็นที่พักข้อมูลชั่วคราว เป็นตัวชี้ข้อมูล และเป็นตัวนับ ส่วนของโมโตโรล่าใช้เป็นอินเด็กซ์ร่วมกับค่าฐาน ซึ่งของซีพียูจำลองตามรูปที่ 2 เป็นแบบผสม ยังมีรีจิสเตอร์อีกกลุ่มหนึ่ง เช่น PC (Program Counter) เป็นตัวชี้แอดเดรสของคำสั่งที่จะปฏิบัติ ซึ่งในการเขียนโปรแกรมย่อยเพื่อใช้กับภาษาสูงยังไม่ต้องสนใจมากนัก และ SP (Stack Pointer) เป็นตัวชี้ชั้นบนสุดของสแต็ค ซึ่งจะนำไปกล่าวในเรื่องสแต็ค

### แฟล็ก

แฟล็ก(flag) หมายถึงรีจิสเตอร์ 1 บิท เพื่อเก็บสถานะ 1 และ 0 ซึ่งสถานะนี้จะนำไปตีความดังจะกล่าวต่อไป ในซีพียูจะมีแฟล็กอยู่หลายตัวรวมอยู่ในรีจิสเตอร์ตัวหนึ่ง เช่น รีจิสเตอร์ F ในรูปที่ 2 และเรียกรีจิสเตอร์ตัวนี้ว่า PSW (Program Status Word) แต่นิยมเรียกกันสั้น ๆ ว่า Flags พอมาถึงเมืองไทยกลายเป็น แฟล็ก ทำให้ผู้เริ่มศึกษางุนงงพอสมควร

ผมจึงได้แยกมาเขียนต่างหาก สำหรับซีพียูจำลองของผมมีแฟล็กอยู่เพียง 2 ตัวเท่านั้น คือแฟล็ก C (Carry flag) ซึ่งจะเซทคือมีค่าเป็น 1 เมื่อมีการทดเนื่องจากการบวก และเมื่อมีการยืมเนื่องจากการลบ เช่น $A3 +$85 = $128 จะได้ค่าในแอคคิวฯเป็น $28 และแฟล็ก C มีค่าเป็น 1 ถ้าหากไม่มีการทดหรือยืมจะทำให้แฟล็ก C เป็นเคลียร์คือมีค่าเป็น 0 ส่วนอีกแฟล็กคือ แฟล็ก Z (Zero flag) ถ้าผลของการดำเนินกรรมวิธีทำให้ค่าในแอคคิวฯเป็น $00 จะทำให้แฟล็ก Z เป็น 1 นอกนั้นเป็น 0 หากคุณสงสัยว่าทำไมถึงกลับกัน ก็เพราะมันชื่อ Zero flag และผมถือว่า 1 หมายถึงเป็นจริง 0 เป็นเท็จ

แฟล็กเป็นเรื่องสำคัญมากเพราะเราจะนำไปเป็นเงื่อนไขในการกระโดด ดังจะเห็นในโปรแกรมตัวอย่าง ดังนั้นในการศึกษาซีพียูเบอร์หนึ่ง ๆ เพื่อการเขียนโปรแกรมจะต้องศึกษาว่ามีแฟล็กใดบ้าง และมีคำสั่งใดบ้างที่ก่อให้เกิดการเปลี่ยนแปลงสถานะใน

แฟลีกใด สุดท้ายคือมีคำสั่งกระโดดตามเงื่อนไขในแฟลีกใดบ้าง

**สแต็ค**

ความจริงแล้วสแต็คเป็นหน่วยความจำหน่วยหนึ่ง แต่ที่นำมาไว้ในเรื่องซีพียูเพราะเป็นพื้นที่ใช้งานของซีพียู ผู้ออกแบบซีพียูส่วนมากจะให้ผู้ใช้คือผู้เขียนโปรแกรมเป็นผู้กำหนดว่า จะให้สแต็คอยู่ที่แอดเดรสใดถึงแอดเดรสใด การกำหนดว่าให้สแต็คเริ่มที่แอดเดรสใดทำโดยให้ค่าแอดเดรสนั้นแก่ SP

ซีพียูจะทำงานกับสแต็คในแบบ FILO (first in last out) เช่นเดียวกับที่เราวางหนังสือซ้อนกัน เมื่อหยิบออกเล่มที่วางเป็นเล่มหลังสุดจะถูกหยิบออกก่อน ส่วนเล่มที่วางเป็นเล่มแรกจะถูกหยิบออกหลังสุด การทำงานของซีพียูกับหน่วยความจำที่เป็นสแต็คมีลักษณะเช่นเดียวกันกับที่กล่าวมานี้ จึงเรียกหน่วยความจำนี้ว่า สแต็ค(ชั้นที่วางของ)

การใช้สแต็คของซีพียูส่วนมากจะทำอยู่ 2 ลักษณะคือ โดยอัตโนมัติและโดยคำสั่งตรง โดยอัตโนมัติคือเมื่อถูกอินเตอร์รัพท์หรือเมื่อปฏิบัติตามคำสั่งเรียกใช้โปรแกรมย่อย เช่น CALL ซึ่งผมจะขอกล่าวเฉพาะเมื่อเรียกใช้โปรแกรมย่อย ซีพียูจะเก็บ คือ push ค่า PC และรีจิสเตอร์อื่น ซึ่งเป็นเรื่องของซีพียูแต่ละเบอร์ลงสแต็ค แล้วกระโดดไปยังแอดเดรสต้นของโปรแกรมย่อยนั้น

ท้ายของโปรแกรมย่อยเราต้องให้คำสั่งส่งกลับ เช่น RET (อ่านว่า return) ซึ่งซีพียูจะถอน คือ pop ค่าในสแต็คคืนให้แก่ PC (และรีจิสเตอร์อื่น ๆ) ทำให้ซีพียูกลับไปทำงานที่คำสั่งต่อจากคำสั่ง CALL ถ้าไม่เข้าใจก็ไม่เป็นไรครับขอให้รับทราบไว้ก็เพียงพอ และที่ผมไม่นำเรื่องของอินเตอร์รัพท์มากล่าว เพราะจะไปเกี่ยวข้องกับฮาร์ดแวร์

อีกลักษณะหนึ่งคือโดยคำสั่งตรง เช่น คำสั่ง PUSH และ POP ค่าในรีจิสเตอร์ จุดประสงค์แรกคือ เพื่อเก็บค่าในรีจิสเตอร์เพื่อนำรีจิสเตอร์ไปใช้งานอื่นก่อน อีกจุดประสงค์หนึ่งคือเพื่อการผ่านค่าไปให้แก่โปรแกรมย่อย ดังจะแสดงในโปรแกรมตัวอย่าง

**ชุดคำสั่ง**

คำสั่งที่เรียกว่า นิวมอนิกส์ (Newmonics) คือคำสั่งที่เป็นคำย่อ ซึ่งของซีพียู 2 ตระกูลใหญ่คือ อินเทลกับโมโตโรลามีลักษณะที่แตกต่างกันอย่างเด่นชัดคือ ของโมโตโรลามีลักษณะเป็น

```
LDA srce
STA dstn
```

ซึ่งอ่านว่า โหลดเอ และ สโตร์เอ ตามลำดับ คำสั่งแรกหมายถึงให้นำค่าจาก srce คือต้นทาง(source) ไปให้แก่แอคคิวฯ คำสั่งหลังหมายถึงให้นำค่าในแอคคิวฯ A ไปให้แก่ปลายทาง(destination) ทั้ง srce และ dstn เป็นหน่วยความจำซึ่งต้องขึ้นกับแบบของการอ้างแอดเดรส

ถ้าเป็นของอินเทลจะมีลักษณะดังแสดงข้างล่างนี้ ซึ่งอ่านว่า มูฟ หมายถึง ให้นำค่าจาก srce คือต้นทางไปให้แก่ dstn คือปลายทาง ซึ่งทั้ง srce และ dstn จะเป็นรีจิสเตอร์หรือหน่วยความจำก็ได้ แต่ยังไม่มีคำสั่งใดที่เป็นหน่วยความจำทั้งคู่ และถ้าเป็นหน่วยความจำจะขึ้นกับแบบของการอ้างแอดเดรสเช่นกัน

```
MOV dstn, srce
```

ผมแสดงนิวมอนิกส์ให้ดูเพื่อจะกล่าวถึงชุดคำสั่งต่อไป คือในซีพียูแต่ละเบอร์จะมีคำสั่งดังกล่าวนี้จำนวนหนึ่งเรียกว่าชุดคำสั่ง ผมไม่เคยทำสถิติว่าคำสั่งในชุดคำสั่งของแต่ละเบอร์มีเท่าไร แต่อาจกล่าวได้ว่าประมาณ 50 คำสั่งขึ้นไป แต่ชุดคำสั่งที่สมบูรณ์ส่วนมากจะทำเป็นตารางชุดคำสั่ง ประกอบด้วยนิวมอนิกส์และออปโค้ด (OP code : operation code) ซึ่งบางแห่งเรียกว่าแมชีนโค้ด (machine code) ดังแสดงของซีพียูจำลองเป็นตัวอย่างในรูปที่ 3 และออปโค้ดนี้คือคำสั่งเครื่องที่แท้จริง หรือคือที่ซีพียูรู้จัก คำสั่งที่เป็นนิวมอนิกส์ก็ดี หรือที่เป็นคำสั่งในภาษาสูงต่าง ๆ ซีพียูไม่รู้จัก กำหนดขึ้นมาเพื่อความสะดวกของมนุษย์เท่านั้น

| OP code | Mnemonics | Addressing mode |
|---|---|---|
| 3E pp | MOV addr,data8 | direct,immediate |
| BE pp qq | MOV addr,data16 | direct,immediate |
| 2C pp qq | MOV A,addr | reg,direct |
| AC pp qq | MOV addr,A | direct,reg |
| 3C pp qq | MOV HL,addr | reg,direct |
| BC pp qq | MOV addr,HL | direct,reg |
| EE 2C pp qq | MOV A,(addr) | reg,indirect |
| EE AC pp qq | MOV (addr),A | indirect,reg |
| 1F pp qq | SUB HL,data16 | reg,immediate |
| 2F pp qq | INC addr8 | direct |
| AF pp qq | INC addr16 | direct |
| 3F pp | CMP addr,data8 | direct,immediate |
| BF pp qq | CMP addr,data16 | direct,immediate |
| 75 ee | BCS disp | relative |
| CD pp qq | CALL addr | direct |
| F3 | PUSH HL | register |

**รูปที่ 3 ชุดคำสั่งของซีพียูจำลอง เฉพาะที่จะใช้ในโปรแกรม**

pp และ qq เป็นเลขฐานสิบหก 2 หลัก เช่น $CO เขียนเป็น CO หากเป็น 4 หลักให้เขียนสลับไบท์เช่น $CO25 เขียนเป็น 25 CO

ee เป็นเลขฐานสิบหก 2 หลักที่คิดเครื่องหมาย

**แบบของการอ้างแอดเดรส**

ในการที่เราจะอ้างถึงหรือบ่งว่าให้ซีพียูติดต่อกับหน่วยความจำ ณ ตำบลที่ใดหรือก็คือไบท์ใดนั้น ซีพียูแต่ละเบอร์จะมีแบบของการอ้างแอดเดรส (Addressing Mode) ไว้ให้เราเลือกใช้ แต่ทั้งนี้ไม่ได้หมายความว่าทุกคำสั่งจะอ้างแอดเดรสได้ทุกแบบ และซีพียูแต่ละเบอร์จะมีแบบของการอ้างแอดเดรสต่างกัน

ที่ผมจะกล่าวต่อไปนี้เป็นแบบการอ้างแอดเดรสของซีพียูจำลอง ซึ่งไม่ได้นำมาจากเบอร์ใดเบอร์หนึ่งโดยเฉพาะ แต่นำมาจาก

หลาย ๆ เบอร์เพื่อให้เห็นแบบของการอ้างแอดเดรสแบบหลัก ๆ หลาย ๆ แบบคือ

1. Immediate เป็นการอ้างแอดเดรสถึงหน่วยความจำที่อยู่ต่อท้ายคำสั่งนั้นเอง เช่น MOV A, #$25 หรือ MOV HL, #$8540 หมายถึงเป็นการให้ค่า $25 แก่ A และ $8540 แก่ HL

2. Direct เป็นการอ้างแอดเดรสถึงหน่วยความจำโดยตรง โดยให้ค่าแอดเดรสไว้ต่อท้ายคำสั่ง เช่น MOV A, $2000 หมายถึงให้ไปนำค่าในหน่วยความจำที่แอดเดรส $2000 มาให้แก่ A

3. Indirect เป็นการอ้างแอดเดรสถึงหน่วยความจำหน่วยหนึ่ง โดยอ้างแอดเดรสถึงหน่วยความจำอีกหน่วยหนึ่ง ซึ่งเก็บแอดเดรสถึงหน่วยความจำที่ต้องการ เช่น MOV A, ($2000) ซีพียูจะต้องไปอ่านค่าที่แอดเดรส $2000 กับ $2001 ว่าเก็บค่าใดไว้ และค่านี้คือแอดเดรสถึงหน่วยความจำที่ต้องการ (แอดเดรสจะเป็น 2 ไบท์ สำหรับเครื่อง 8 บิท)

**โปรดสังเกตว่า ในการเขียนนิวมอนิกส์ทั้ง 3 แบบนี้ เราเขียนต่างกันและเมื่อแปลงเป็นออปโค้ด เช่นในเรื่องของ MOV A, srce ก็จะแตกต่างกันด้วย*

4. Register Indirect เป็นการอ้างแอดเดรสถึงหน่วยความจำผ่านรีจิสเตอร์ คือต้องให้ค่าแอดเดรสแก่รีจิสเตอร์ก่อน เช่น MOV HL, #$2000 แล้วอ้างแอดเดรสผ่านรีจิสเตอร์นี้ เช่น MOV A,(HL) จะเป็นการนำค่าในแอดเดรส $2000 มาให้แก่ A

**เพื่อความเข้าใจเพิ่ม สมมติว่าเราตั้งชื่อหน่วยความจำที่แอดเดรส $2000 กับ $2001 ว่า PNTR (การตั้งชื่อให้หน่วยความจำเป็นเรื่องที่นิยมปฏิบัติกันในการเขียนโปรแกรมภาษาเครื่อง) เราจะอ้างแอดเดรสได้ดังนี้*

```
MOV HL, =PNTR ; immediate
MOV A, (HL)   ; register indirect
MOV A, (PNTR) ; indirect
```

*เครื่องหมาย ; หมายความว่าข้อความหลังเครื่องหมายนี้เป็นหมายเหตุซึ่งใช้กันมากในภาษาเครื่อง หากคุณสงสัยว่ามี indirect แล้วทำไมจึงต้องมี register indirect ก็เพราะผมจับมารวมกัน ซีพียูที่ผมรู้จักจะมีเพียงแบบใดแบบหนึ่งใน 2 แบบนี้เท่านั้น*

5. Base Indexed Indirect คล้ายกับ Indirect แต่ให้นำค่าที่ได้จาก Indirect มาเป็นฐาน คือมาบวกกับค่าในอินเด็กซ์รีจิสเตอร์ เช่น IX เป็นแอดเดรสถึงหน่วยความจำที่ต้องการ เช่น MOV A,(PNTR)+IX

6. Relative เป็นการอ้างแอดเดรสสัมพันธ์กับ PC โดยต้องให้ค่าสัมพันธ์เป็นเลขเครื่องหมาย 8 บิท เพื่อนำไปบวกกับค่าใน PC ในแบบพีชคณิต

แบบของการอ้างแอดเดรสที่ผมนำมากล่าวนี้เป็นเพียงหลัก ๆ แบบของการอ้างแอดเดรสของซีพียูจริงจะแตกย่อยออกไปอีก และไม่เหมือนกันในแต่ละเบอร์ นอกจากนี้ยังอาจเรียกชื่อไม่ตรงกับที่กล่าวมา เช่น ซีพียูที่ไม่มี Indirect อาจเรียก Register Indirect ว่าเป็น Indirect สำหรับชื่อที่ผมกล่าวถึงนี้หากเรียกเต็ม ๆ เช่น Immediate ต้องเรียกเป็น Immediate Addressing Mode

## ก่อนจบตอน

เท่าที่กล่าวมานี้ก็เพื่อให้ทราบถึงเรื่องราวของภาษาเครื่องอย่างกว้าง ๆ และเป็นแนวทางในการทำความเข้าใจซีพียูแต่ละเบอร์ ซึ่งผมเห็นว่าการศึกษาซีพียูเพียงเพื่อเขียนโปรแกรมย่อยให้แก่ภาษาสูงนั้น การศึกษาในเรื่องที่ผมกล่าวมานี้น่าจะเพียงพอ และควรศึกษาตามลำดับดังนี้

– ได้จัดรีจิสเตอร์ไว้อย่างไร แต่ละตัวมีหน้าที่พิเศษอะไรบ้าง รีจิสเตอร์ที่เก็บแฟล็กมีแฟล็กอะไรบ้าง

– แบบของการอ้างแอดเดรสได้กำหนดไว้อย่างไรบ้าง

– ในเรื่องของชุดคำสั่ง ในขั้นแรกนี้ให้สนใจเฉพาะคำสั่งในเรื่อง การเคลื่อนย้ายข้อมูล การคำนวณ การกระโดดและการเรียกโปรแกรมย่อย ซึ่งถ้ามีตารางชุดคำสั่งที่แบ่งหมวดหมู่ของคำสั่งไว้จะช่วยได้มาก จึงให้ดูที่ตารางนี้ก่อน การศึกษาแต่ละคำสั่งให้ศึกษาว่า ใช้กับแบบของการอ้างแอดเดรสใดได้บ้าง เมื่อปฏิบัติตามคำสั่งนี้แล้วจะเกิดผลอะไรขึ้น และมีผลกระทบต่อแฟล็กอะไรบ้างอย่างไร (ควรสนใจเฉพาะแฟล็ก C กับแฟล็ก Z ก่อน)

ขอให้ทดลองศึกษาของซีพียูในเครื่องของคุณก่อน ในตอนหน้าเมื่อผมกล่าวถึงการเขียนโปรแกรมภาษาเครื่องคงจะเข้าใจชัดเจนขึ้นครับ ถ้าคุณเห็นว่ามันยากก็ขอเรียนให้ทราบว่าเป็นเพราะมันใหม่ต่อคุณ เมื่อคุ้นเคยและไม่พยายามจะสงสัยในเรื่องที่ลึกซึ้งก็จะเป็นเรื่องง่าย ได้แสดงรายชื่อหนังสือที่จะใช้ซึ่งก็เป็นแต่เพียงคำแนะนำเท่านั้น ไว้ท้ายบทความนี้ด้วย

**หนังสือ**

หนังสือสำหรับซีพียูในเครื่องไมโครคอมพิวเตอร์ที่ใช้กันมากในบ้านเราได้แก่ Z-80, 6502, 8085 และ 8088 คือ

1. An Introduction to Microcomputer Volume 2 Some Real Microprocessor ของ osboene เป็นหนังสือรวมซีพียูหลายเบอร์รวมทั้ง 4 เบอร์ที่กล่าวด้วย หนังสือเล่มนี้ถึงแม้จะหนักทางด้านฮาร์ดแวร์ แต่ก็กล่าวถึงเรื่องราวที่ต้องการทั้งหมด

2. หนังสือในชุดของ Lance A. Leventhal เช่น 8085 Assembly Language Programming กล่าวถึงชุดคำสั่งและการใช้คำสั่งแต่ละคำสั่งอย่างละเอียด มีโปรแกรมของโครงงานตัวอย่างหลายโครงงาน และมีที่ตั้งชื่อเป็น 6502 Assembly Language Subroutines สำหรับของ Z80 ผมไม่แน่ใจว่ามีหรือไม่

3. หนังสือของ Z-80 มีอยู่มาก ให้เลือกเล่มที่ถูกใจเพียงเล่มเดียวก็พอ ที่ขอแนะนำคือ Z-80 Users Manual โดย Joseph J. carr ซึ่งเขียนในเรื่องที่ต้องการโดยตรง แต่ไม่มีตัวอย่างอยู่เลย

4. The 8086 Book (include the 8088) โดย Russell Rector and George Alesey กล่าวถึงชุดคำสั่งโดยละเอียดและมีเรื่องทางฮาร์ดแวร์ด้วย

5. โปรแกรมคอมพิวเตอร์ภาษาแอสเซมบลี 8086/8088 โดย ยืน ภู่วรวรรณ และสุรศักดิ์ สงวนพงษ์ กล่าวถึงชุดคำสั่งตลอดจนการใช้แอสเซมเบอร์และการดีบัก มีตัวอย่างประกอบ

6. วารสารเซมิคอนดัคเตอร์อิเล็กทรอนิกส์ ได้ลงบทความ "แฟ้ม CPU" กล่าวถึงซีพียูเบอร์ต่าง ๆ ตั้งแต่ฉบับที่ 66 ดังนี้ เบอร์ 8085 ฉบับที่ 66, เบอร์ 6502 ฉบับที่ 67, เบอร์ 6809 ฉบับที่ 68, เบอร์ 8086/8088 ฉบับที่ 70 และมีเรื่องของ Z80 ตั้งแต่ฉบับที่ 65 ถึง 67